Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 26 (2015) 397 – 402

www.elsevier.com/locate/procedia

12th Global Conference on Sustainable Manufacturing

Tool Path Generation, for Complex Surface Machining, Using Point Cloud Data

Anadil Masood*, Rooha Siddiqui, Michelle Pinto, Hira Rehman, Dr. Maqsood A. Khan

NED University of Engineering and Technology, University Road, Karachi-75270, Pakistan

* Corresponding author. Tel.: +923468063723; *E-mail address:* anadil_masood@yahoo.com

Abstract

This paper gives a detailed explanation of the project, related to reverse engineering, which was conducted by the authors of this paper. The project was based on direct machining, which is done by generating efficient tool paths directly from point cloud data, stored in STL format. The primary objective was to achieve high efficiency in the machining of free-form surface geometries, having complex machining areas. Reverse engineering traditionally involves surface fitting. However, it has several drawbacks. To overcome these drawbacks, the concept of direct machining is used. This skips the surface fitting process and consists of three main steps: digitizing, tool path generation and machining. Therefore, an algorithm to generate tool paths for direct machining was developed. The algorithm works for three-axis milling, using a ball-end mill cutter. It includes dividing the surface into ranges and generating B-spline curves which are best fit curves within each range. These curves are the required tool paths. A few case studies have also been conducted using this algorithm.

Peer-review under responsibility of Assembly Technology and Factory Management/Technische Universität Berlin.

Keywords: reverse engineering; direct machining; point cloud data; STL format; free-form surfaces; B-Spline Curves; algorithm

1. Introduction

In today's manufacturing world, complex surfaces are being used in various areas of work; such as to create medical replicas and in automotive and aerospace industries. The punches, dies and molds for creating these parts through processes like sheet metal forming, forging and casting also need to have complex geometries. In the past, work has been done in the field of reverse engineering (RE). Traditionally, RE involves three steps; (1) acquisition of discretized data of the surface, using a 3D scanner, in the form of point cloud (2) surface reconstruction i.e. converting the data into CAD files and (3) tool path generation. This means conversion of acquired data from one format to the other, twice. Hence, the chances of error being introduced increase. This is because when the data is converted to another form, approximation takes place, depending on the level of accuracy of the other format. Deviation from the original object during RE is not acceptable in many cases. This led to the idea of direct machining. A Z-map model [1] was used to create a grid and iso-planar tool paths were generated by adjusting the tool vertically, to avoid gouging. For finishing operations, smooth segments were linked by making sure each segment overlapped the previous segment. This was the first reported work on direct machining. Later, to avoid time

Peer-review under responsibility of Assembly Technology and Factory Management/Technische Universität Berlin.
doi:10.1016/j.procir.2014.07.076

consumption during calculation of the cutter location (CL) surface, 2D curve offsetting and polygon chain intersections were used to generate finishing tool paths [2]. After which, work was done on generating CL paths from triangular mesh [3,4]. With the advancement in triangulation of data points, Stereolithography (STL) files are also being used. Moreover, a CL-net has also been developed [5] and CL points were generated using weighted averages of CL-net node positions [6]. CL-net is identified by forward steps and side steps.

Recently, algorithms for direct machining are being generated via two methods. Area by area machining [7, 8] is one of them. In this method, very complex portions of the surface are separated from lesser complex portions and a smaller or larger tool is respectively selected. The second method involves the use of planes that intersect the surface, to achieve the same purpose. The parallel plane strategy has been used on a Z-map model for machining [9]. Drive planes have also been used to intersect the mean least square (MLS) surface [10]. The resulting intersecting curves were the cutter contact (CC) curves. The developed tool paths were linear i.e. splines were not used. MLS surface is defined by the local minimum of an energy function and normal vectors. The spatial points are then projected on the MLS surface. MLS was initially proposed by Levin [11]. MLS can handle noise, up sample and down sample [12]. The work using MLS surface continued and an additional semi-finish cut was introduced, using the CL method [13]. Later, CL points, that lay on the drive planes, were used for tool path generation. This was achieved by using, the proposed B-Spline Curve differential equations, for interpolation [14].

This project includes the development of an algorithm that takes in data obtained by using a 3 dimensional scanner. A 3 D scan must be taken in order to obtain the digital data of the object that needs to be duplicated. There are many non-contact scanners that now work using laser beams. These mostly use electromagnetic radiations. This is called optical digitizing. When a 3D scan of any object is taken using a 3D scanner, whether it is a contact scanner such as a Coordinate Measuring Machine (CMM) [15] or an optical scanner like ATOS [16] or a FARO Arm, the data obtained is in the form of many points on its surface. This data is called point cloud data. The data in this project was collected using the laser scanner of a 6 axis, 4 ft Faro Arm, having an accuracy of ±0.018mm. The laser scanner in this case is a Model Maker X35 having the wavelength of 635.660 nm and power of 5 MW. The collected data was stored in STL format. In this format the data is stored in triangulated form. The algorithm divides the triangulated surfaces into ranges. The division of each range is similar to intersection planes. This algorithm takes in point cloud data in STL format and generates tool paths without converting it into CAD files. The difference in this algorithm from the previous researches [9, 10, 13 and 14] is that here tool paths are generated using B-spline interpolation, which is a best fit curve within each range. This means that more points within a range can be taken into consideration while generating tool paths. This is because best fit curves takes into consideration all the points in a locality whether or not they lie on the curve.

2. Explaining the Developed Algorithm

An algorithm was developed using MATLAB. The basic data element in MATLAB is a matrix. Furthermore, in MATLAB, functions can be created. These are small programs that take in inputs from the main program and give output as per the requirements of the main program. These functions make the program understandable and compact.

2.1. Import STL file

The STL file used stores data in ASCII format. This data needs to be imported into MATLAB and it acts as input to the program. There are various functions available for this purpose. The mode in the function used, gives vertices of the triangles and their respective normal. These normal vectors are normal to each triangle and correspond to the centroids of their respective triangle.

2.2. Calculate centroids

The algorithm is equipped to calculate the x, y and z coordinates of the centroid of each triangle so that the position of the normal vector of the triangle is known. The centroids are the CC points. This is important for further calculation, explained in the paper in section 2.3.

2.3. Sorting, converting to CL points and dividing into ranges

The x values were arranged in ascending order, using an inbuilt command in MATLAB; ‘sort’. This was done so that the values of x could easily be divided into ranges.

The instructions given to the machine for tooling are done according to the CL points. The CC points lie on the surface of the part or the object being machined and the cutters comes in contact with these points while machining. This is explained in figure 1 and 2. The following formula was used to convert CC points to CL points:

$$\mathbf{CL = CC + rn} \quad (1)$$

Where **r** is the radius of a ball-end mill cutter and **n** is the normal vector on the CC point.

Once this was achieved, the CL points were divided into ranges with respect to x coordinates. Each range is to have a separate tool. This is because if there is a tool path for every value of x then there will be infinite number of tool paths. This will be practically impossible for machining.

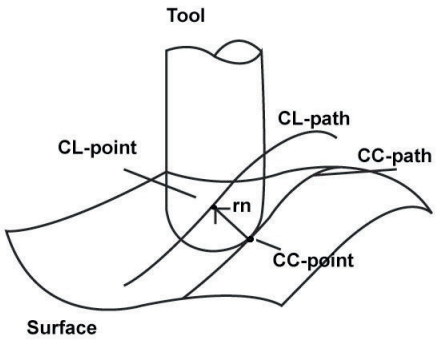

Figure 1: Explaining CC and CL points

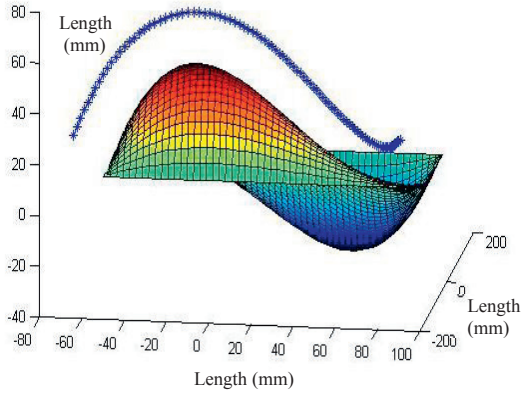

Figure 2: Path followed by the center of the tool

2.4 Arranging the CL points in a sequence

STL files store triangulated data in a random fashion, which means; the points will also be plotted randomly. The CL points need to be arranged in a sequence, since the tool needs to follow a sequential path. This is illustrated in figure 3 and figure 4.

A built- in command in MATLAB, called **Unique,** was used. This arranges data in an ascending order and removes any repetitive value. This command works for each range separately. The surface is divided into ranges with respect to the x values. **Unique** arranges the data within a range in an ascending order with respect to y and omits any y value that is repeating itself in that range. In this way, the points in a range are organized in sequence, moving in an increasing fashion with respect to y.

The problem with using the command; **Unique**, is that the y values that are repeating themselves will be omitted and only one of each similar value will remain.

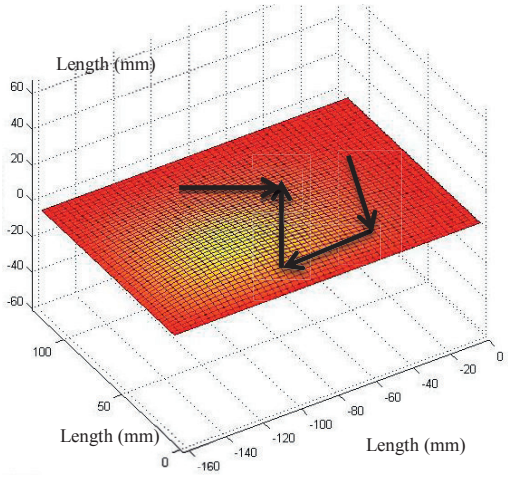

Figure 3: CL points in random order

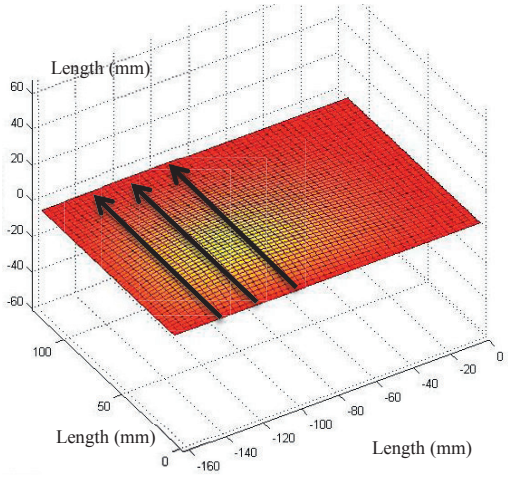

Figure 4: CL points in sequence

This means that some of the points from the STL file would be eliminated. If a great number of points are omitted, a large number of data will be lost. If that happened, the main purpose of the project, which is to reduce error, would not be achieved.

To avoid this problem, the width of the ranges was reduced by increasing the number of ranges. When the ranges were made narrower, the y values that were repeating themselves earlier, within a range, were now in the next range. Points repeating themselves, within a range were reduced, reducing the number of points being omitted. So the original

data did not change greatly. These points were then plotted.

2.5. Saving the values

The plotted points needed to be stored for further reference and work. The values were stored in a single array containing x, y and z values of all the ranges.

After this, if-else looping was used. This looping required that the x, y and z values be greater than zero that is, that the surface should be in the positive axes overall. This was done because zeros being introduced due to which each spline, after its completion, kept going back to the origin. Upon studying, it was realized that the zeros were introduced when the x values were divided into ranges. This was because the values of x were stored in a matrix. Each range had different number of values. To fill the rows, which stored the x values of a range, zeros were added. The corresponding y and z values also became zero. Hence, when the command; 'unique', was applied it ignored all the zeros except for one. The extra zeros being introduced needed to be eliminated for which the looping was used.

2.6. Spline interpolation

For spline interpolation a function; **spap2**, was used. This function requires four input arguments; knot vectors, order of the spline, parametric values and data points. It then gave the path to be followed by the tool, i.e. it gave the spline which is actually the cutter location points.

2.6.1. Knot vectors

A function is used to calculate knot vectors. This function gives an output in the form of knot vectors which is used as an input for spap2.

2.6.2. Order

Order is calculated using the following formula.

Order = degree+1. (2)

Degree, in this case, is 3. Hence, order = 4.

2.6.3. Calculating chord lengths for parametric values

Chord length needed to be calculated, to calculate parameters. These parametric values are needed for the function '**spap2**'. Chord lengths are calculated by adding the previous length of chord to the new length and dividing it by the sum of chord lengths for all the points. This is shown in figure 5. Each chord length added to all the previous chord lengths, divided by the total length gives the parametric values.

$$D=d_1+d_2+d_3\ldots..+d_n \quad (3)$$

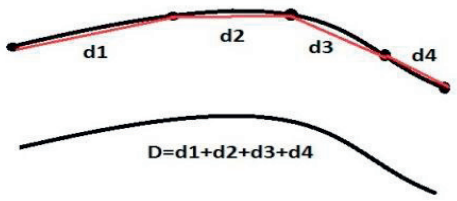

Figure 5: Showing chord lengths and their sum

2.6.4 Data points

These are the points that were calculated using the command; 'unique'.

2.7 Summary of the project

This section summarizes the description of the proposed algorithm, with the help of figure 6.

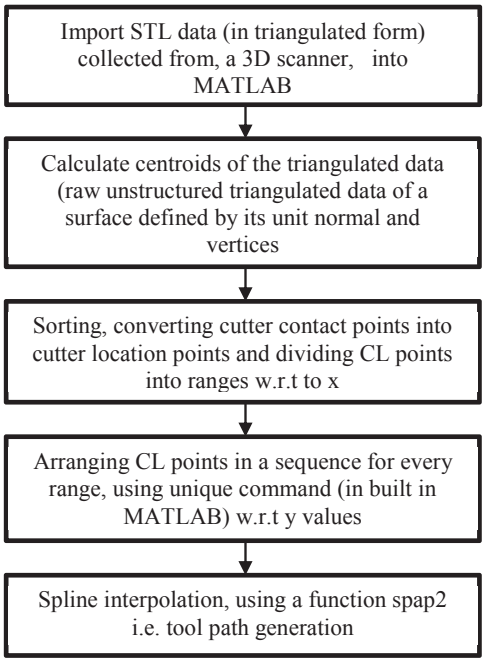

Figure 6: Summary of the proposed algorithm

3. What the Algorithm Does and Its Outcomes

The algorithm plots the actual surface, the points that are calculated using the unique command and the splines, that are the generated tool paths. It also stores the points calculated using the command; 'unique' and the splines. Figure 7 and 8 show **Case A** with five and fifteen splines, respectively, generated for the first surface. It also shows the points in each range for which the spline was generated.

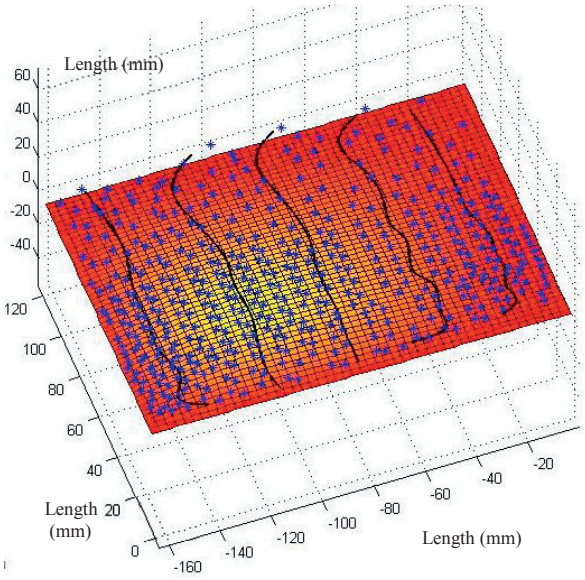

Figure 7: Case A; 5 splines generated for a surface

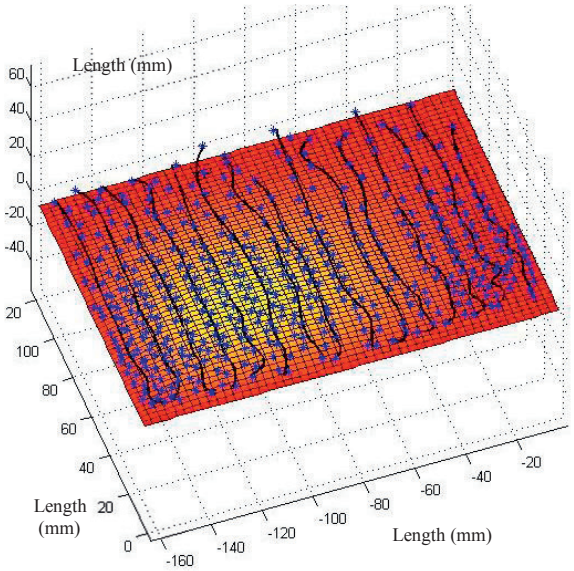

Figure 8: Case A; 15 splines generated for a surface

The first surface tested was of the dimension: 163mm in the x direction and 113mm in the y direction. It was initially divided into 5 ranges. Since the ranges are in the x direction, the width of each range was 32.6mm. The surface was then divided into 15 ranges i.e. the width of the each range now was 10.9 mm.

Using the algorithm the number of splines generated can be controlled and as a result the scallop height is controlled. This affects the surface finish. Scallop height and surface finish are two measurements that are used for characterizing surfaces, particularly when the surface is created using a cutting tool, as shown in figure 9. The surface finish can be analysed using specialized equipment like a light or camera that scans the surface.

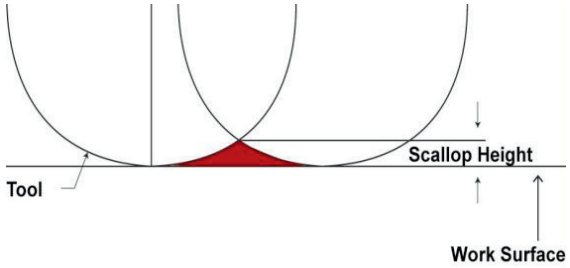

Figure 9: Showing scallop height

4. Testing the Algorithm on Different Surfaces

The robustness of the developed algorithm was tested on different surfaces that had higher accuracy and modified geometry. This was done by using higher resolution of the scanner or by using a starting surface that has a different and more complex geometry, respectively. Splines for each case were then generated and the paths along which the cutter will move were observed.

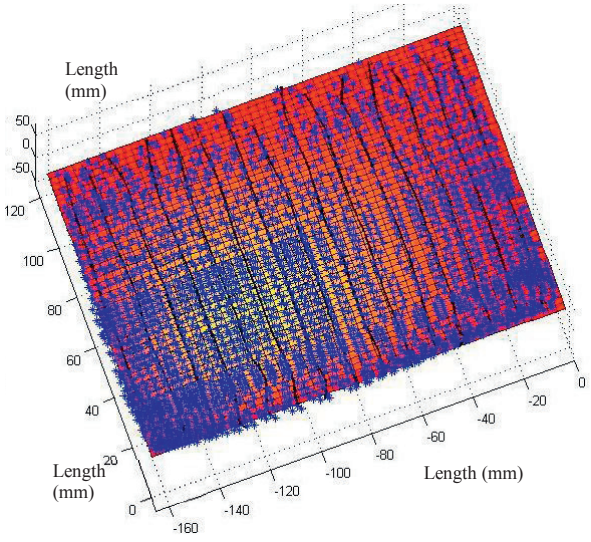

Figure 10: Case study B; maximum level of accuracy for the first surface

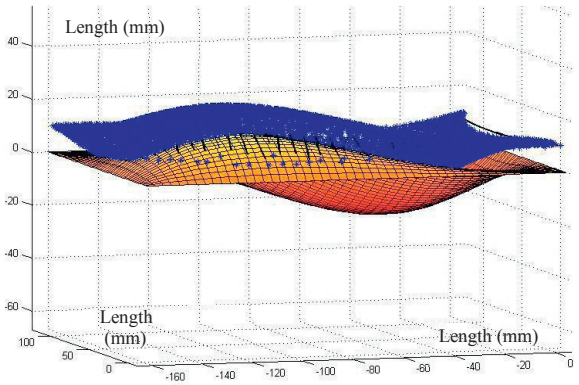

Figure 11: Case Study C (a); modified geometry

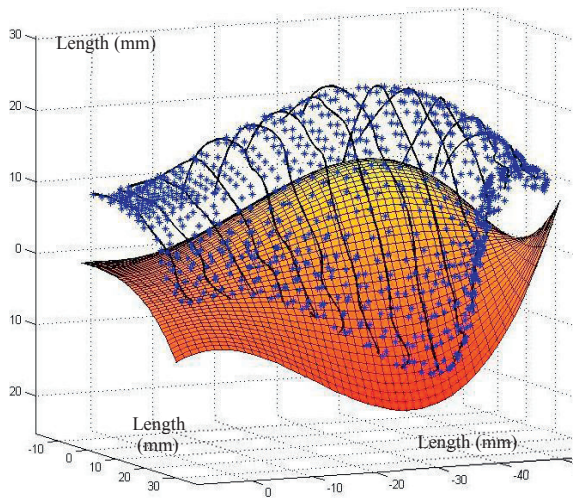

Figure 12: Case Study C (b): modified geometry

Figure 10 shows that the algorithm is suitable when the accuracy of the digitized data is increased. Figures 11 and 12 show that the algorithm is also suitable for more complex surface.

5. Possible Further Work

The program is very robust when it comes to generating tool paths for different surfaces that are aligned with the x and y axes. However, the program is not equipped to handle surfaces that have any other shape at the edges other than a rectangle. Hence, the algorithm can be modified to include functions or commands through which tool paths can be generated for surfaces having other shapes as well.

Many 3D scanners have the option of shifting the origin and aligning the axes according to the part being scanned. However, this algorithm can be modified by incorporating transformation matrices. This will make it easier for the users to scan and generate tool paths, without worrying about the alignment of the surface or adjusting the origin.

6. Acknowledgements

We would like to thank Mr. Danish Mahmood Khan (Sr. Lab Engineer, Telecommunications Department, NED University of Engineering and Technology, Karachi) and Dr. Muhammad Wasif (Assistant Professor, Industrial and Manufacturing Department, NED University of Engineering and Technology, Karachi) for their willingness to answer any queries we had and for supporting us whenever we approached them.

7. References

[1] Alan C. Lin and Hai-Terng Liu, 1998, 'Automatic Generation of NC Cutter Path from Massive Data Points', Computer Aided Design, vol. 30, no.1, pp. 77-90.

[2] Sang. C. Park and Yun. C. Chung, 2003, 'Tool-path Generation from Measured Data', Computer-Aided Design, vol. 35, no. 1, pp. 467- 475.

[3] Sang C. Park, September 2004, 'Triangular Mesh Intersection', The Visual Computer, vol. 20, issue 7, pp. 448-456.

[4] Lee, S. H., Kim, H. C., Hur, S. M., and Yang, D. Y.,2002, 'STL File Generation From Measured Point Data by Segmentation and Delaunay Triangulation', Computer-Aided Design, pp. 691-704.

[5] His-Yung Feng, and Zhengji Teng, 2005, 'Iso-Plannar Piecewise Linear NC Tool Path Generation from Discrete Measured Data Points, Computer-Aided Design', vol. 37, pp. 55-64.

[6] Daoshan OuYang1 and Hsi-Yung Feng, 2008, 'Machining Triangular Mesh Surfaces via Mesh Offset Based Tool Paths', Computer-Aided Design and Applications, CAD Solutions, LLC, pp 254-265.

[7] Mohanad Makki, Christophe Tournier, François Thiebaut, Claire Lartigue and Charyar Souzani, 2010, '5-axis Direct Machining of Rough Clouds of Points', Computer Aided Design, vol. 7(4), pp. 591-600.

[8] M. Makkia, C. Lartiguea, C. Tourniera, F. Thiébauta, 2011, 'Direct Duplication of Physical Models in Discrete 5-axis Machining', Virtual and Physical Prototyping 3, 2, Version 1, pp. 93-103.

[9] Z. Tchantchane, M. Bey, 2009, 'Toward the Roughing of Sculptured Surfaces from a Regular Cloud of Points', 13th International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology" TMT 2009, Hammamet, Tunisia, pp. 21-24.

[10] Dongdong Zhang, Pinghai Yang, Xiaoping Qian, 2009**, '**Adaptive NC Path Generation from Massive Point Data with Bounded Error', Journal of manufacturing Science and Engineering, ASME, vol. 131.

[11] David Levin, October 1998, 'The Approximation Power of Moving Least-Squares,' Mathematics of Computation, Volume 67, number 224, pages 1517-1531.

[12] Tamal K. Dey and Jian Sun, 2005, "An Adaptive MLS Surface for Reconstruction with Guarantees", Eurographics Symposium on Geometry Processing.

[13] Zixian Zhang, Maria Savchenko, Ichiro Hagiwara, and BingyinRen, 2010, '3-Axis NC Tool Path Generation and Machining Simulation for Subdivision Surface of Complex Models', International Journal of CAD/CAM, vol. 10, no. 1, pp. 1-9.

[14] Yu Liu, Songtao Xia, Xiaoping Qian, 2012, 'Direct NC Path Generation: from Discrete Points to Continuous Spline Paths', Journal of Computing and Information Science in Engineering, ASME, vol. 12, Issue 3.

[15] Manzoor Hussain M., Sambasiva Rao CH. and Prasad K. E., August 2008, 'Reverse Engineering: Point Cloud Generation with CMM for Part Modeling and Error Analysis", ARPN Journal of Engineering and Applied Sciences, VOL. 3, NO. 4.

[16] Galanulis, K, Reich, C., Thesing, J., Winter, D., 2010, Optical Digitization by ATOS for Press Parts and Tools, GOM Gesellschaft für Optische Messtechnik mbH, Braunschweig.